Projet 5 : Recherche tags StackOverFlow

Philippine KLING

Sommaire

PARTIE 1 : récupérer les données

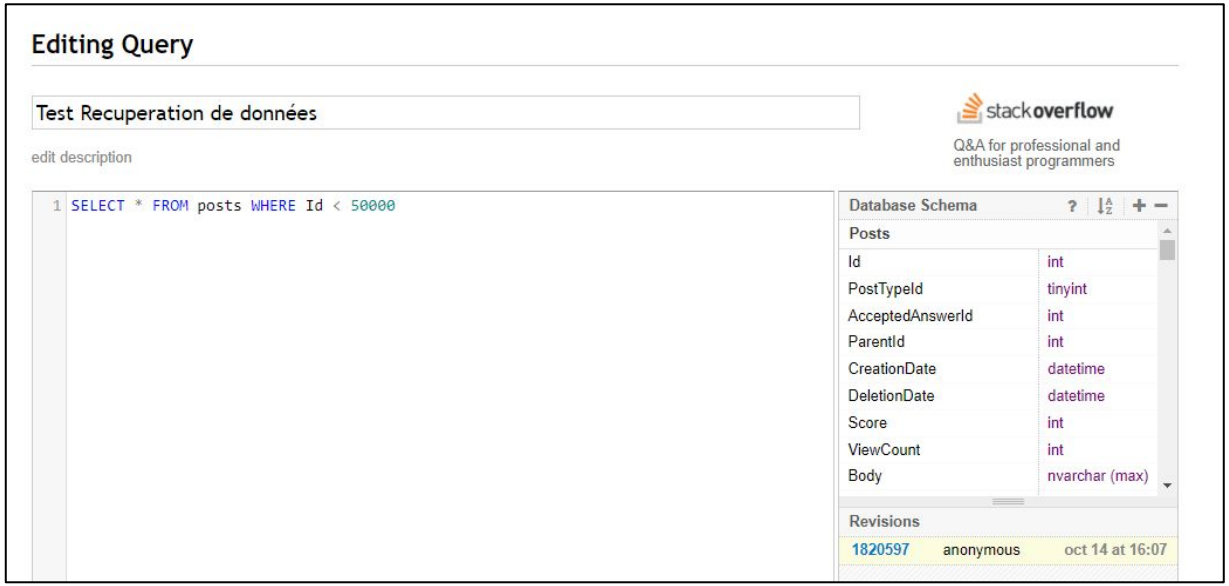
PARTIE 2 : EDA / Analyse exploratoire

PARTIE 3 : Prétraitement des données

PARTIE 4 : entrainer les modèles qui vont prédire les tags

PARTIE 5 : présenter les résultats sous forme d'une API

PARTIE 1 : récupérer les données



présentation de la plateforme de récupération des données de stackoverflow

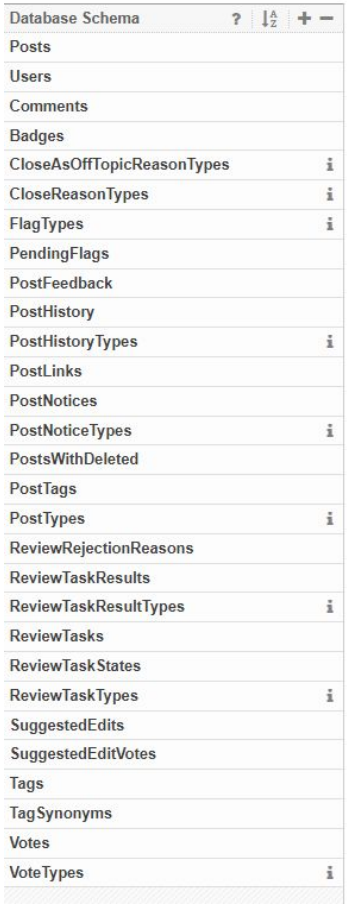
PARTIE 1 : récupérer les données

Récupération des questions :

1. SELECT Id, OwnerUserId, CreationDate, ClosedDate, Score, Title, Body FROM posts
2. WHERE Id < 50000
3. AND OwnerUserId is not Null
4. AND Score > 0
5. AND Title is not Null

Récupération des Tags

1. SELECT PostId, TagName
2. FROM PostTags left join Tags on TagId = Tags.Id
3. WHERE PostId < 50000



liste des bibliothèques présentes sur la base de données

PARTIE 2 : EDA / Analyse exploratoire

Présentation du fichier des questions

Id	OwnerUserId	CreationDate	ClosedDate	Score	Title	Body
4	8	2008-07-31 21:42:52	NaN	752	How to convert a Decimal to a Double in C#?	<p>I want to use a <code>Track-Bar</code> to c...
6	9	2008-07-31 22:08:08	NaN	312	Why did the width collapse in the percentage w...	<p>I have an absolutely positioned <code>div</...
9	1	2008-07-31 23:40:59	NaN	2083	How do I calculate someone's age based on a Da...	<p>Given a <code>DateTime</code> representing ...
11	1	2008-07-31 23:55:37	NaN	1599	Calculate relative time in C#	<p>Given a specific <code>DateTime</code> valu...
13	9	2008-08-01 00:42:38	NaN	667	Determine a user's timezone	<p>Is there a standard way for a web server to...

PARTIE 2 : EDA / Analyse exploratoire

	Score
count	5031.000000
mean	70.814550
std	315.624477
min	1.000000
25%	4.500000
50%	11.000000
75%	33.000000
max	7318.000000

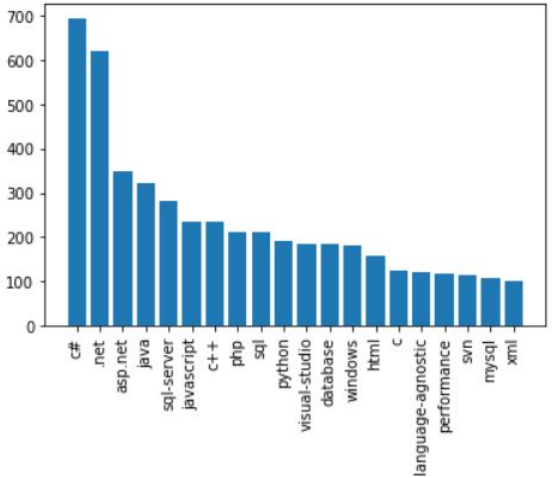
description du score

```
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Id            5031 non-null   int64
 1   OwnerUserId   5031 non-null   int64
 2   CreationDate  5031 non-null   object
 3   ClosedDate    704 non-null    object
 4   Score         5031 non-null   int64
 5   Title         5031 non-null   object
 6   Body          5031 non-null   object
dtypes: int64(3), object(4)
memory usage: 275.3+ KB
```

information générales sur la table

PARTIE 2 : EDA / Analyse exploratoire

Most frequent tags



Répartitions des différents tags

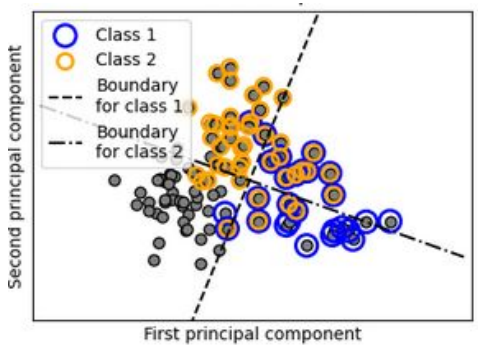
PARTIE 3 : Prétraitement des données

```
def nettoyage_text(text):
    text = retire_html(text)
    text = retire_punck(text)
    text = lemitize_words(text)
    text = retire_stopwords(text)
    return text
```

Préparation des données pour la classification

PARTIE 4 : entrainer les modèles qui vont prédire les tags

OneVsRest Classifier



PARTIE 4 : entrainer les modèles qui vont prédire les tags

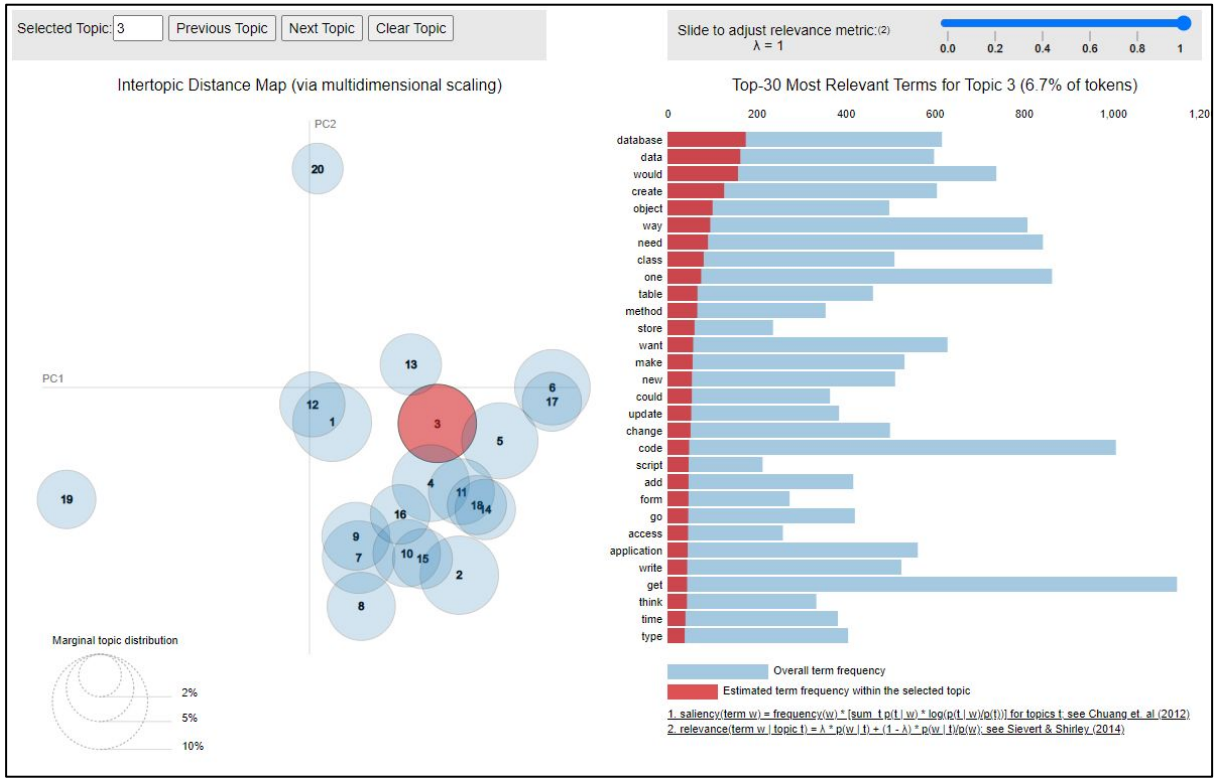
	Classifier	Jacard_avg	Jacard_macro	Hamming	Accuracy	Recall	precision	Time	f1
0	OneVsRestClassifier	0.060561	0.032848	0.123412	0.017725	0.063432	0.061175	0 days 00:00:00.039032	0.096268
1	OneVsRestClassifier	0.482029	0.429095	0.047046	0.370753	0.491359	0.722980	0 days 00:00:00.090711	0.582295
2	OneVsRestClassifier	0.306992	0.265045	0.049483	0.249631	0.271836	0.818457	0 days 00:00:00.345710	0.440735
3	OneVsRestClassifier	0.077548	0.058747	0.062555	0.064993	0.059452	0.495069	0 days 00:00:00.036861	0.125903
4	OneVsRestClassifier	0.469719	0.418337	0.044904	0.367799	0.460648	0.786326	0 days 00:00:00.127232	0.582418
5	OneVsRestClassifier	0.435106	0.384485	0.064919	0.264402	0.540927	0.538948	0 days 00:00:00.090086	0.520458
6	OneVsRestClassifier	0.448227	0.396025	0.061891	0.292467	0.525118	0.572987	0 days 00:00:00.308482	0.526554
7	MLPClassifier	0.449902	0.400592	0.051551	0.327917	0.464944	0.710217	0 days 00:00:13.230721	0.548512
8	RandomForestClassifier	0.400788	0.313318	0.045790	0.339734	0.323243	0.801339	0 days 00:00:03.595457	0.507937

PARTIE 4 : présenter les résultats sous forme d'une API

exemples de topics et des labels associé

```
.net: represent syntax domain vista native via generic wcf assembly net
asp.net: side section postback control page cache cause web webconfig aspnet
c: level argument include b i gcc char platform c cc
c#: thoughts listview namespace way compact foreach lock core datatable c
c++: linux certain exist bool 64 include const win32 c cc
database: join char trick transaction language record past databases db database
html: tag batch font site give inline put width div html
java: api distribute application happen look action dependency javadoc eclipse java
javascript: function ajax js virtual side browser var ie jquery javascript
language-agnostic: statement regex similar case non prefer function ways based language
mysql: integration in live entry support 3 statements limit query mysql
performance: site efficient scale sense time fastest faster optimize speed performance
php: live specific post integration resize entry apache permissions session php
python: thing plug x none iterate drag parameters equivalent django python
sql: length based time graph choice index select insert join sql
sql-server: insert procedure script ms 2005 master fill temp server sql
svn: source proxy development control merge version branch repository subversion svn
visual-studio: custom textbox ui flag project team 2003 debug studio visual
windows: question user export standard core task thoughts drive software windows
xml: find replace section choice quite comment apply node nod xml
```

Classification LDA



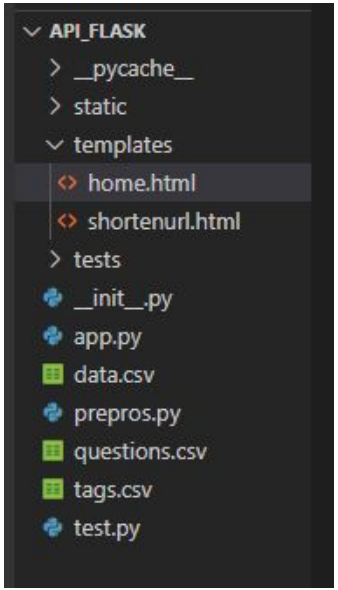
PARTIE 5 : présenter les résultats sous forme d'une API

home.html

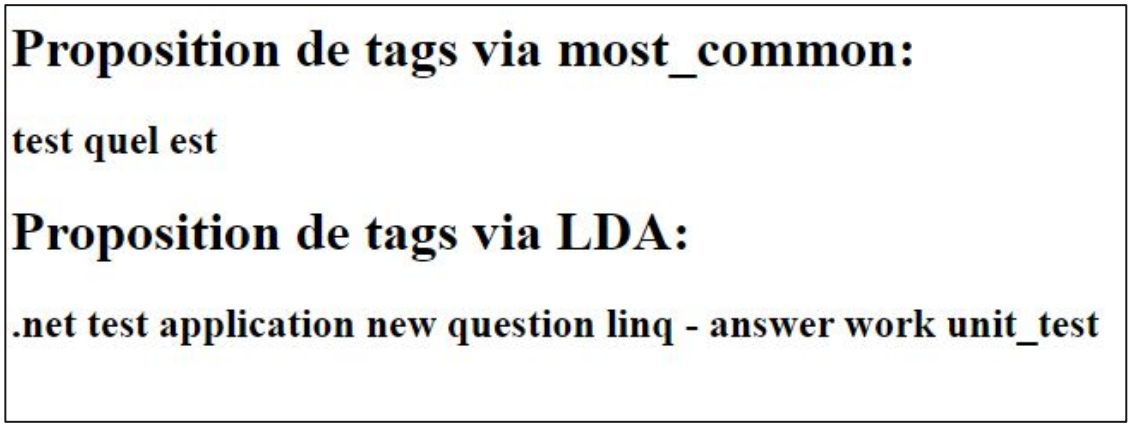
```
templates > home.html > html > body > form
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>API Stack Overflow</title>
</head>
<body>
<h1>Prédiction de tags Stack Overflow</h1>
<form action="shortenurl" method="post">
    <!-- Champ pour le titre -->
    <label for="title">Title</label>
    <input type="text" name="title" value="" required>
    <!-- Champ pour le développement -->
    <label for="question">Question</label>
    <input type="text" name="question" size="100" value="" required>
    <!-- Champ pour soumettre -->
    <input type="submit" value="Submit">
</form>

</body>
</html>
```

shortenurl.html

```
templates > shortenurl.html > ...
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Here Is Your URL!</title>
</head>
<body>
<h1>Proposition de tags  via most_common:</h1>
<h2>{{ shortcode1 }}</h2>
<h1>Proposition de tags  via LDA:</h1>
<h2>{{ shortcode2 }}</h2>
</body>
</html>
```



PARTIE 5 : présenter les résultats sous forme d'une API



test de l'API

Conclusion

Les avantages

- Ce projet m’a permis d’utiliser les bibliothèques de traitement de texte
- La LDA permet de proposer des tags qui ne sont pas forcément dans le texte.

Les pistes d’améliorations

- Il faudrait utiliser plus de données pour la classification LDA de sorte à créer plus de catégories différentes. Par contre cela risque d’augmenter significativement le temps de calcul du modèle
- travailler sur les synonymes des tags
- utiliser les réponses comme élément de la question pour les modèles d'entraînement